UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/727,240	12/03/2003	Jose Abad Peiro	200313161-1	4907

22879 7590 06/26/2006

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
SAIN, GAUTAM	

ART UNIT	PAPER NUMBER
2176	

DATE MAILED: 06/26/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/727,240	PEIRO ET AL.	
	Examiner	Art Unit	
	Gautam Sain	2176	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *03 December 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-34* is/are pending in the application.

4a) Of the above claim(s) *29-31* is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-28 and 32-34* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *12/3/03*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1) This is a non final rejection in response to application filed on 12/3/2003.

2) Claims 1-34 are pending. Claims 1-28 and 32-34 are rejected.

3) Claims 29-31 are withdrawn without traverse (see Election details below).

Election/Restrictions

4) Restriction to one of the following inventions is required under 35 U.S.C. 121:

I. Claims 1-28 and 32-34, drawn to generating a PDF document from a PPLM document, classified in class 715, subclass 513.

II. Claims 29-31, drawn to generating a PPML document from a PDF document, classified in class 715, subclass 526.

4-1) The inventions are distinct, each from the other because of the following reasons:

Inventions I and II are related as subcombinations disclosed as usable together in a single combination. The subcombinations are distinct if they do not overlap in scope and are not obvious variants, and if it is shown that at least one subcombination is separately usable. In the instant case, subcombination I generates a PDF document from a PPLM document, where a structured document is converted to an image document as the final product where as subcombination II generates a PPML document from a PDF document, where the image document is converted to a final product of a structured document. See MPEP § 806.05(d).

Because these inventions are independent or distinct for the reasons given above and have acquired a separate status in the art because of their recognized **divergent subject matter**, restriction for examination purposes as indicated is proper.

Claims 29-31 are withdrawn from further consideration pursuant to 37 CFR 1.142(b) as being drawn to a nonelected invention, there being no allowable generic or linking claim. Election was made **without** traverse by Applicant on a telephone call on 6/19/2006. Applicant (David Thompson) elected claims 1-28 and 32-34 for prosecution in the current application.

Specification

5) The Examiner requests that Applicant review the application carefully for informalities including typographical errors.

The cross reference related to the applications cited int eh specification must be updated (ie., update the relevant status, with PTO serial numbers or patent numbers where appropriate on page 1). Correction is required.

Claim Objections

6) Claims 1-28 and 32-34 are objected to because of the following informalities: the abbreviations (ie., PDF, PPML, XML and XSLT) used in the claims need to be defined. Correction is required.

Claim Rejections - 35 USC § 101

7) 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

7-1) Claims 14-28 and 32 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. Claims 14-28 and 32 set forth functional descriptive material but fail to set forth physical structures or materials comprising of hardware or a combination of hardware and software within the

technological arts (ie., a computer) to produce a "useful, concrete and tangibly" result. Claims 14-28 and 32 are interpreted as software per se, abstract ideas or mental construct and not tangibly embodied on a computer readable medium or hardware.

Claim Rejections - 35 USC § 103

8) The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8-1) Claims 1-10, 12-15, 17-22, 24-28 and 32-34 rejected under 35 U.S.C. 103(a) as being unpatentable over Gebert et al (US 2002/0111963, published on 8/25/2006), in view of Nonpatent Literature PODI Technical Meeting (Nov 2000, available at http://www.info-dist.com/lg/technical/argon/argon_public.pdf>)(hereinafter "PODI").

Regarding claims 1, Gebert teaches the processor-executable instructions comprising instructions for: parsing structures within the PPML document; generating a PDF document tree; and interpreting the parsed structures from the PPML document onto locations on the PDF document tree. For example, Gebert discloses a method and program for preprocessing a document to render on an output device (Gebert, Title) where the program converts XML document into a viewable format such as PDF by using an XML parser to parse the XML result tree and then transform the tree elements into a PDF format including formatting object into DOM documents to access elements of the result tree in the DOM (para 8). Additionally, to transform the XML document, the

entire source file is transformed to a result tree such as formats objects that define an internal representation of the page layouts (para 7).

The Examiner interprets the claimed "generating a PDF document from a PPML document" as equivalent to *generating a PDF document from an XML document*, because PPML is an XML-based language for variable-data printing (see Applicant's specification section, page 1, para 2), thus generating a PDF document from an PPML document is interpreted as requiring the same functionalities as generating a PDF from XML.

Gebert does not expressly teach "*generating a PDF document from a PPML document*", but PODI does teach this limitations. Specifically, PODI discloses a presentation at a Technical Meeting in 2000 by think121.com, where PODI expressly disclosed product plug-ins for converting vendor and device-specific PPML to generic PDF (see presentation slides, slide page 30 as shown on page 20 of the attached reference).

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Gebert to include a plug-ins for converting PPML documents to PDF documents as expressly disclosed by PODI, providing the benefit of efficient and real-time manipulation of underlying PDF document and structures (PODI, slide 5) and providing the user tremendous flexibility in reuse of information (PODI, slide 25 and 26) for a printer (Gebert, para 11), which are similar benefits sought by the Applicant in the invention of allowing users to use lower-end printers that can process PDF or Postscript

documents for print jobs that originated with PPML documents (see Applicant specification, Background section, page 1, para 2 and 3).

Regarding claim 2, Gebert teaches configuring a PDF document according to the PDF document tree; and printing the PDF document. For example, Gebert discloses a program that converts XML document into a viewable format such as PDF by using an XML parser to parse the XML result tree and then transform the tree elements into a PDF format including formatting object into DOM documents to access elements of the result tree in the DOM (para 8). Additionally, to transform the XML document, the entire source file is transformed to a result tree such as formats objects that define an internal representation of the page layouts (para 7) in order to transform the XML document for output on a printer (para 11).

Regarding claim 3, Gebert teaches interpreting the parsed structures comprising instructions for resolving, for objects within the PPML document, a PPML SOURCE_TYPE class; and translating the objects according to the PPML SOURCE_TYPE class. For example, Gebert discloses processing a source XML document to determine the document objects such as page divisions and formatting properties, and then transmitting the page objects to transform the page objects into a readable information capable of being generated by an output device (para 12). The Examiner interprets Gebert's teaching as equivalent to the claimed functionality, which is to interpret the parsed structure, resolving the objects of the document and then translating the objects in order to prepare them for printing.

Regarding claim 4, Gebert teaches un-marshalling a PPML instance, when a PPML tag refers to an external object; and embedding the external object into the PDF document. Gebert teaches an XML parser to parser XML document to parser XML result tree and then transform the tree elements into output in a PDF (para 8), where the transformation may contain text, graphic or images objects to define the page layout for presentation to a reader (para 7). The Examiner interprets the claimed "un-marshalling a PPML" as equivalent to parsing because the specification describe un-marshalling as one of the steps for parsing (see Applicant specification, page 6, para 19).

Regarding claim 5, Gebert teaches wherein the external object to which the PPML tag refers includes fonts and images. Gebert discloses nodes of the result tree including format objects such as fonts and images for rendering on a printer device (para 6 and 7).

Regarding claim 6, Gebert teaches interpreting the parsed structures comprises instructions for: resolving references within the parsed structures into PDF assets; and incorporating the PDF assets into the PDF document. Gebert discloses processing a source XML document to determine the document objects such as page divisions and formatting properties, and then transmitting the page objects to transform the page objects into a readable information capable of being generated by an output device (para 12) for optimal rendering of PDF documents at the printer.

Regarding claim 7, Gebert teaches resolving an image asset into an InputSource object using a PPML structure; and incorporating the image asset into the PDF document. Gebert discloses processing a source XML document to determine the

document objects such as page divisions and formatting properties, and then transmitting the page objects to transform the page objects into a readable information capable of being generated by an output device (para 12) for optimal rendering of PDF documents at the printer nodes of the result tree including format objects such as fonts and images for rendering on a printer device (para 6 and 7).

Regarding claim 8, Gebert suggests that the PPML structure comprises: <SOURCE Format="image/jpeg" Dimensions "dim1 dim2"> <EXTERNAL_DATA_ARRAY Src="filename.jpg"/> </SOURCE>. The Examiner interprets this limitation as processing steps towards translating XML document into PDF document including external image objects. Gebert discloses processing a source XML document to determine the document objects such as page divisions and formatting properties including image data objects (para 7), and then transmitting the page objects to transform the page objects into a readable information capable of being generated by an output device (para 12) for optimal rendering of PDF documents at the printer nodes of the result tree including format objects such as fonts and images for rendering on a printer device (para 6 and 7).

Regarding claim 9, Gebert teaches instructions for resolving assets within the parsed structures into objects; and incorporating the objects into the PDF document. Gebert discloses processing a source XML document to determine the document objects such as page divisions and formatting properties, and then transmitting the page objects to transform the page objects into a readable information capable of being generated by an output device (para 12) for optimal rendering of PDF documents at the printer.

Regarding claim 10, Gebert suggest resolving assets using a PPML structure to translate an asset into an iText font representing a TTF font. The Examiner interprets this claim as using an XML structure to translate fonts in order to have them retain properties for an accurate representation when generating the fonts in a PDF. Gebert teaches using an XML structure (a DOM tree) of nodes for elements of XML source documents to construct the result tree for transformation of objects including fonts (para 6) and for then transforming the tree elements into PDF (para 8).

Regarding claim 12, Gebert suggests interpreting the parsed structures comprises instructions for: locating PPML global impositions when parsing the structures within the PPML document; and reutilizing the PPML global impositions. The Examiner interprets the claimed impositions as properties for a template that define how logical pages will be mapped into physical pages and may be input as parameters by user (see Applicant's specification, page 15, para 47). Gebert discloses classes of formatting objects and formatting properties (para 12) for transforming XML documents to output that is rasterized to a PDF (para 7). The XSL formatting objects generates from source XML document and defines the page layout for presentation to a reader (para 12). The examiner interprets this as reutilized because the source objects and nodes are reused for creating a resultant node tree or layout.

Regarding claim 13, Gebert does not expressly teach instructions for reutilizing IMPOSITION_TYPE instances, but does suggest it because Gebert discloses classes of formatting objects and formatting properties (para 12) for transforming XML documents to output that is rasterized to a PDF (para 7). The XSL formatting objects

generates from source XML document and defines the page layout for presentation to a reader (para 12).

It would have been obvious to one of ordinary skill in the art at the time of the invention to interpret Gebert's teaching of formatting objects and formatting properties for transforming XML documents to output that is rasterized to a PDF as equivalent to the claimed reutilizing instance because the source objects and nodes are reused for creating a resultant node tree or layout and this limitation would be an obvious step in the implementation of the transformation and layout process.

Regarding claim 14, Gebert teaches a first component to parse structures and tag items within the PPML document; a second component to generate a PDF document tree; and a third component to translate the tagged items within the PPML document, and to locate the translated items on the PDF document tree. For example, Gebert discloses a method and program for preprocessing a document to render on an output device (Gebert, Title) where the program converts XML document into a viewable format such as PDF by using an XML parser to parse the XML result tree and then transform the tree elements into a PDF format including formatting object into DOM documents to access elements of the result tree in the DOM (para 8). Additionally, to transform the XML document, the entire source file is transformed to a result tree such as formats objects that define an internal representation of the page layouts (para 7).

The Examiner interprets the claimed "generating a PDF document from a PPML document" as equivalent to *generating a PDF document from an XML document*, because PPML is an XML-based language for variable-data printing (see Applicant's

specification section, page 1, para 2), thus generating a PDF document from an PPML document is interpreted as requiring the same functionalities as generating a PDF from XML.

Gebert does not expressly teach "converting a PPML document into a PDF document", but PODI does teach this limitations. Specifically, PODI discloses a presentation at a Technical Meeting in 2000 by think121.com, where PODI expressly disclosed product plug-ins for converting vendor and device-specific PPML to generic PDF (see presentation slides, slide page 30 as shown on page 20 of the attached reference).

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Gebert to include a plug-ins for converting PPML documents to PDF documents as expressly disclosed by PODI, providing the benefit of efficient and real-time manipulation of underlying PDF document and structures (PODI, slide 5) and providing the user tremendous flexibility in reuse of information (PODI, slide 25 and 26) for a printer (Gebert, para 11), which are similar benefits sought by the Applicant in the invention of allowing users to use lower-end printers that can process PDF or Postscript documents for print jobs that originated with PPML documents (see Applicant specification, Background section, page 1, para 2 and 3).

Regarding claim 15, Gebert teaches a forth component to un-marshal PPML instances of external objects for embedding within the PDF document tree. Gebert teaches an XML parser to parser XML document to parser XML result tree and then transform the tree elements into output in a PDF (para 8), where the transformation may contain text,

graphic or images objects to define the page layout for presentation to a reader (para 7). The Examiner interprets the claimed "un-marshalling a PPML" as equivalent to parsing because the specification describe un-marshalling as one of the steps for parsing (see Applicant specification, page 6, para 19).

Regarding claim 17, Gebert teaches resolving PPML SOURCE_TYPE class of the structures within the PPML document. For example, Gebert discloses processing a source XML document to determine the document objects such as page divisions and formatting properties, and then transmitting the page objects to transform the page objects into a readable information capable of being generated by an output device (para 12). The Examiner interprets Gebert's teaching as equivalent to the claimed functionality, which is to interpret the parsed structure, resolving the objects of the document and then translating the objects in order to prepare them for printing.

Regarding claim 18, Gebert suggests the component translating the items tagged as a function of a type associated with each of the items tagged. Gebert teaches transforming source XML DOM tree into a result DOM tree structure to allowing printing PDF documents, where the source tree and result tree have classes of formatting objects information such as page, paragraph, table, font, etc., ... (para 6). The examiner interprets these classes of formatting information as equivalent to the claimed tagged items of a type, because the Examiner interprets tags as details about objects in order to classify the data object into a category/class for transforming a particular class and Gebert's formatting objects such as fonts can be of a font type of formatting object information.

Regarding claim 19, Gebert suggests translating a PDF object within a PdfTemplate defined as a function of an item type of the tagged items found during parsing by the first component. Gebert teaches transforming source XML DOM tree into a result DOM tree structure to allowing printing PDF documents, where the source tree and result tree have classes of formatting objects information such as page, paragraph, table, font, etc., (para 6). The examiner interprets these classes of formatting information as equivalent to the claimed tagged items of a type, because the Examiner interprets tags as details about objects in order to classify the data object into a category/class for transforming a particular class and Gebert's formatting objects such as fonts can be of a font type of formatting object information. Additionally, Gebert discloses the XML parser to parse the XML formatting objects result tree and then transform the tree elements into a PDF document

Regarding claim 20, Gebert teaches wherein the PdfTemplate is configured to support caching of objects to optimize PDF structure. Gebert discloses optimizing output for rendering on a specific printer (para 10).

Regarding claim 21, Gebert teaches means for parsing structures within the PPML document; means for generating a PDF document tree; and means for interpreting the parsed structures from the PPML document onto locations on the PDF document tree. For example, Gebert discloses a method and program for preprocessing a document to render on an output device (Gebert, Title) where the program converts XML document into a viewable format such as PDF by using an XML parser to parse the XML result tree and then transform the tree elements into a PDF format including formatting object

into DOM documents to access elements of the result tree in the DOM (para 8). Additionally, to transform the XML document, the entire source file is transformed to a result tree such as formats objects that define an internal representation of the page layouts (para 7).

The Examiner interprets the claimed "generating a PDF document from a PPML document" as equivalent to *generating a PDF document from an XML document*, because PPML is an XML-based language for variable-data printing (see Applicant's specification section, page 1, para 2), thus generating a PDF document from an PPML document is interpreted as requiring the same functionalities as generating a PDF from XML.

Gebert does not expressly teach "converting a PPML document into a PDF document", but PODI does teach this limitations. Specifically, PODI discloses a presentation at a Technical Meeting in 2000 by think121.com, where PODI expressly disclosed product plug-ins for converting vendor and device-specific PPML to generic PDF (see presentation slides, slide page 30 as shown on page 20 of the attached reference).

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Gebert to include a plug-ins for converting PPML documents to PDF documents as expressly disclosed by PODI, providing the benefit of efficient and real-time manipulation of underlying PDF document and structures (PODI, slide 5) and providing the user tremendous flexibility in reuse of information (PODI, slide 25 and 26) for a printer (Gebert, para 11), which are similar benefits sought by the Applicant in the

invention of allowing users to use lower-end printers that can process PDF or Postscript documents for print jobs that originated with PPML documents (see Applicant specification, Background section, page 1, para 2 and 3).

Regarding claim 22, Gebert teaches means for un-marshalling a PPML instance, when a PPML tag set by the means for parsing refers to an external object; and means for embedding the external object into the PDF document. Gebert teaches an XML parser to parser XML document to parser XML result tree and then transform the tree elements into output in a PDF (para 8), where the transformation may contain text, graphic or images objects to define the page layout for presentation to a reader (para 7). The Examiner interprets the claimed "un-marshalling a PPML" as equivalent to parsing because the specification describe un-marshalling as one of the steps for parsing (see Applicant specification, page 6, para 19).

Regarding claim 24, Gebert teaches means for resolving, for objects within the PPML document, a PPML SOURCE_TYPE class; and means for translating the objects according to the PPML SOURCE_TYPE class. For example, Gebert discloses processing a source XML document to determine the document objects such as page divisions and formatting properties, and then transmitting the page objects to transform the page objects into a readable information capable of being generated by an output device (para 12). The Examiner interprets Gebert's teaching as equivalent to the claimed functionality, which is to interpret the parsed structure, resolving the objects of the document and then translating the objects in order to prepare them for printing.

Regarding claim 25, Gebert teaches parsing structures within the PPML document; generating a PDF document tree; and interpreting the parsed structures from the PPML document onto locations on the PDF document tree. For example, Gebert discloses a method and program for preprocessing a document to render on an output device (Gebert, Title) where the program converts XML document into a viewable format such as PDF by using an XML parser to parse the XML result tree and then transform the tree elements into a PDF format including formatting object into DOM documents to access elements of the result tree in the DOM (para 8). Additionally, to transform the XML document, the entire source file is transformed to a result tree such as formats objects that define an internal representation of the page layouts (para 7).

The Examiner interprets the claimed "generating a PDF document from a PPML document" as equivalent to *generating a PDF document from an XML document*, because PPML is an XML-based language for variable-data printing (see Applicant's specification section, page 1, para 2), thus generating a PDF document from an PPML document is interpreted as requiring the same functionalities as generating a PDF from XML.

Gebert does not expressly teach "converting a PPML document into a PDF document", but PODI does teach this limitations. Specifically, PODI discloses a presentation at a Technical Meeting in 2000 by think121.com, where PODI expressly disclosed product plug-ins for converting vendor and device-specific PPML to generic PDF (see presentation slides, slide page 30 as shown on page 20 of the attached reference).

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Gebert to include a plug-ins for converting PPML documents to PDF documents as expressly disclosed by PODI, providing the benefit of efficient and real-time manipulation of underlying PDF document and structures (PODI, slide 5) and providing the user tremendous flexibility in reuse of information (PODI, slide 25 and 26) for a printer (Gebert, para 11), which are similar benefits sought by the Applicant in the invention of allowing users to use lower-end printers that can process PDF or Postscript documents for print jobs that originated with PPML documents (see Applicant specification, Background section, page 1, para 2 and 3).

Regarding claim 26, Gebert teaches means for un-marshalling a PPML instance, when a PPML tag set during parsing refers to an external object; and embedding the external object into the PDF document. Gebert teaches an XML parser to parser XML document to parser XML result tree and then transform the tree elements into output in a PDF (para 8), where the transformation may contain text, graphic or images objects to define the page layout for presentation to a reader (para 7). The Examiner interprets the claimed "un-marshalling a PPML" as equivalent to parsing because the specification describe un-marshalling as one of the steps for parsing (see Applicant specification, page 6, para 19).

Regarding claim 27, Gebert teaches interpreting the parsed structures comprises instructions for: resolving references within the parsed structures into PDF assets; and incorporating the PDF assets into the PDF document. Gebert discloses processing a source XML document to determine the document objects such as page divisions and

formatting properties, and then transmitting the page objects to transform the page objects into a readable information capable of being generated by an output device (para 12) for optimal rendering of PDF documents at the printer.

Regarding claim 28, Gebert teaches resolving an image asset into an InputSource object using a PPML structure; and incorporating the image asset into the PDF document. Gebert discloses processing a source XML document to determine the document objects such as page divisions and formatting properties, and then transmitting the page objects to transform the page objects into a readable information capable of being generated by an output device (para 12) for optimal rendering of PDF documents at the printer nodes of the result tree including format objects such as fonts and images for rendering on a printer device (para 6 and 7).

Regarding claim 32, Gebert teaches A method for printing a PDF file, comprising: sending a PPML file to a PPML to PDF converter; converting the PPML file into the PDF file, wherein the converting comprises parsing and tagging items within the PPML file and translating the tagged items; and printing the PDF file. For example, Gebert discloses a method and program for preprocessing a document to render on an output device (Gebert, Title) where the program converts XML document into a viewable format such as PDF by using an XML parser to parse the XML result tree and then transform the tree elements into a PDF format including formatting object into DOM documents to access elements of the result tree in the DOM (para 8). Additionally, to transform the XML document, the entire source file is transformed to a result tree such as formats objects that define an internal representation of the page layouts (para 7).

The Examiner interprets the claimed "generating a PDF document from a PPML document" as equivalent to *generating a PDF document from an XML document*, because PPML is an XML-based language for variable-data printing (see Applicant's specification section, page 1, para 2), thus generating a PDF document from an PPML document is interpreted as requiring the same functionalities as generating a PDF from XML.

Gebert does not expressly teach converting a PPML document into a PDF document, but PODI does teach this concept. Specifically, PODI discloses a presentation at a Technical Meeting in 2000 by think121.com, where PODI expressly disclosed product plug-ins for converting vendor and device-specific PPML to generic PDF (see presentation slides, slide page 30 as shown on page 20 of the attached reference).

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Gebert to include a plug-ins for converting PPML documents to PDF documents as expressly disclosed by PODI, providing the benefit of efficient and real-time manipulation of underlying PDF document and structures (PODI, slide 5) and providing the user tremendous flexibility in reuse of information (PODI, slide 25 and 26) for a printer (Gebert, para 11), which are similar benefits sought by the Applicant in the invention of allowing users to use lower-end printers that can process PDF or Postscript documents for print jobs that originated with PPML documents (see Applicant specification, Background section, page 1, para 2 and 3).

Regarding claim 33, Gebert teaches A system for printing a PDF file, comprising: a PPML to PDF converter, configured to receive a PPML file and configured to create the PDF file; and a printing device, to print the PDF file. For example, Gebert discloses a method and program for preprocessing a document to render on an output device (Gebert, Title) where the program converts XML document into a viewable format such as PDF by using an XML parser to parse the XML result tree and then transform the tree elements into a PDF format including formatting object into DOM documents to access elements of the result tree in the DOM (para 8). Additionally, to transform the XML document, the entire source file is transformed to a result tree such as formats objects that define an internal representation of the page layouts (para 7).

The Examiner interprets the claimed "generating a PDF document from a PPML document" as equivalent to *generating a PDF document from an XML document*, because PPML is an XML-based language for variable-data printing (see Applicant's specification section, page 1, para 2), thus generating a PDF document from an PPML document is interpreted as requiring the same functionalities as generating a PDF from XML.

Gebert does not expressly teach converting a PPML document into a PDF document, but PODI does teach this concept. Specifically, PODI discloses a presentation at a Technical Meeting in 2000 by think121.com, where PODI expressly disclosed product plug-ins for converting vendor and device-specific PPML to generic PDF (see presentation slides, slide page 30 as shown on page 20 of the attached reference).

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Gebert to include a plug-ins for converting PPML documents to PDF documents as expressly disclosed by PODI, providing the benefit of efficient and real-time manipulation of underlying PDF document and structures (PODI, slide 5) and providing the user tremendous flexibility in reuse of information (PODI, slide 25 and 26) for a printer (Gebert, para 11), which are similar benefits sought by the Applicant in the invention of allowing users to use lower-end printers that can process PDF or Postscript documents for print jobs that originated with PPML documents (see Applicant specification, Background section, page 1, para 2 and 3).

Regarding claim 34, Gebert teaches a parsing and tagging component configured to parse the PPML file and locate references to assets; a translation component, to translate the assets located by the references; and a PDF tree-generating component, to generate a PDF tree to form the PDF file according to the translated assets. For example, Gebert discloses a method and program for preprocessing a document to render on an output device (Gebert, Title) where the program converts XML document into a viewable format such as PDF by using an XML parser to parse the XML result tree and then transform the tree elements into a PDF format including formatting object into DOM documents to access elements of the result tree in the DOM (para 8). Additionally, to transform the XML document, the entire source file is transformed to a result tree such as formats objects that define an internal representation of the page layouts (para 7).

The Examiner interprets the claimed "generating a PDF document from a PPML document" as equivalent to *generating a PDF document from an XML document*, because PPML is an XML-based language for variable-data printing (see Applicant's specification section, page 1, para 2), thus generating a PDF document from an PPML document is interpreted as requiring the same functionalities as generating a PDF from XML.

Gebert does not expressly teach converting a PPML document into a PDF document, but PODI does teach this concept. Specifically, PODI discloses a presentation at a Technical Meeting in 2000 by think121.com, where PODI expressly disclosed product plug-ins for converting vendor and device-specific PPML to generic PDF (see presentation slides, slide page 30 as shown on page 20 of the attached reference).

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Gebert to include a plug-ins for converting PPML documents to PDF documents as expressly disclosed by PODI, providing the benefit of efficient and real-time manipulation of underlying PDF document and structures (PODI, slide 5) and providing the user tremendous flexibility in reuse of information (PODI, slide 25 and 26) for a printer (Gebert, para 11), which are similar benefits sought by the Applicant in the invention of allowing users to use lower-end printers that can process PDF or Postscript documents for print jobs that originated with PPML documents (see Applicant specification, Background section, page 1, para 2 and 3).

8-2) Claims 11, 16 and 23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Gebert (as cited above), in view of PODI (as cited above), further in view of Nonpatent Literature "The PPML Print Language in XML Workflows for Digital Print" (May 2001 by Dave deBronkart, available at <http://www.gca.org/papers/xmleurope2001/papers/html/sid-03-5.html>)(hereinafter "deBronkart").

Regarding claim 11, Gebert in view of PODI does not teach, but deBronkart suggests interpreting the parsed structures comprises instructions for reutilizing PPML global objects found within the PPML document with a PPML structure, wherein the PPML global objects are configured as REUSABLE_OBJECT_TYPE. DeBronkart teaches a PPML print language in XML workflow for digital print (Title), where personalized documents with reusable content are largely mapped onto conventional workflows wherein the variable data records trigger the selection of reusable objects, using logic rules embedded in a template for generating PDF content objects using a PDF generating tool (page 3, section 3.3).

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Gebert in view of PODI to include reusable content objects are mapped to conventional workflows with variable data records for generating PDF content objects from PPML objects as taught by deBronkart, providing the benefit of promoting interoperability amongst printers allowing users to have a fast time to market (deBronkart, page 3, section 4) for variable data printing applications, allowing users to use desktop printers (deBronkart, Abstract), which is similar to the goal of the

Applicant's invention of allowing users to use any printer such as low-end printers instead of high-end digital presses (see specification, page 1, para 3).

Regarding claim 16, Gebert teaches a SourceResolver to resolve references into assets for attachment to the PDF document tree. The examiner interprets this limitations as a system to resolve objects with the PDF document tree, including image objects. Gebert teaches an XML parser to parser XML document to parser XML result tree and then transform the tree elements into output in a PDF (para 8), where the transformation may contain text, graphic or images objects to define the page layout for presentation to a reader (para 7). The result document (ie., PDF document) includes root elements and page sequence elements such as text and images (para 24). The resultant PDF document is in a DOM tree of nodes of elements with rearranged source data objects and added information necessary to format the result tree for presentation, upon completion of a tree transformation (para 6).

Gebert teaches a FontResolver Interface to resolve fonts for access by the PDF document. Gebert discloses a result PDF document from the transformation of the source XML document with nodes of element objects including fonts (para 6). The fonts in the XML source document are transformed over to the resultant PDF document. Gebert suggests an ImpositioningStore to reutilize PPML global impositions for access by the PDF document. The Examiner interprets the limitation of impositions as properties for a template that define how logical pages will be mapped into physical pages and may be input as parameters by user (see Applicant's specification, page 15, para 47). Gebert discloses classes of formatting objects and formatting properties

(para 12) for transforming XML documents to output that is rasterized to a PDF (para 7). The XSL formatting objects generates from source XML document and defines the page layout for presentation to a reader (para 7 and 12). The examiner interprets this as reutilized because the source objects and nodes are reused for creating a resultant node tree or layout.

Gebert in view of PODI does not teach, but deBronkart suggests an OccurrenceStore Interface to reutilize PPML global objects for access by the PDF document. DeBronkart teaches a PPML print language in XML workflow for digital print (Title), where personalized documents with reusable content are largely mapped onto conventional workflows wherein the variable data records trigger the selection of reusable objects, using logic rules embedded in a template for generating PDF content objects using a PDF generating tool (page 3, section 3.3).

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Gebert in view of PODI to include reusable content objects are mapped to conventional workflows with variable data records for generating PDF content objects from PPML objects as taught by deBronkart, providing the benefit of promoting interoperability amongst printers allowing users to have a fast time to market (deBronkart, page 3, section 4) for variable data printing applications, allowing users to use desktop printers (deBronkart, Abstract), which is similar to the goal of the Applicant's invention of allowing users to use any printer such as low-end printers instead of high-end digital presses (see specification, page 1, para 3).

Regarding claim 23, Gebert teaches a means for resolving references into assets for attachment to the PDF document tree. The examiner interprets this limitations as a system to resolve objects with the PDF document tree, including image objects. Gebert teaches an XML parser to parser XML document to parser XML result tree and then transform the tree elements into output in a PDF (para 8), where the transformation may contain text, graphic or images objects to define the page layout for presentation to a reader (para 7). The result document (ie., PDF document) includes root elements and page sequence elements such as text and images (para 24). The resultant PDF document is in a DOM tree of nodes of elements with rearranged source data objects and added information necessary to format the result tree for presentation, upon completion of a tree transformation (para 6).

Gebert teaches a means for resolving fonts for access by the PDF document. Gebert discloses a result PDF document from the transformation of the source XML document with nodes of element objects including fonts (para 6). The fonts in the XML source document are transformed over to the resultant PDF document.

Gebert suggests means for reutilizing PPML global impositions for access by the PDF document. The Examiner interprets the limitation of impositions as properties for a template that define how logical pages will be mapped into physical pages and may be input as parameters by user (see Applicant's specification, page 15, para 47). Gebert discloses classes of formatting objects and formatting properties (para 12) for transforming XML documents to output that is rasterized to a PDF (para 7). The XSL formatting objects generates from source XML document and defines the page layout

for presentation to a reader (para 7 and 12). The examiner interprets this as reutilized because the source objects and nodes are reused for creating a resultant node tree or layout.

Gebert in view of PODI does not teach, but deBronkart suggests means for reutilizing PPML global objects for access by the PDF document. DeBronkart teaches a PPML print language in XML workflow for digital print (Title), where personalized documents with reusable content are largely mapped onto conventional workflows wherein the variable data records trigger the selection of reusable objects, using logic rules embedded in a template for generating PDF content objects using a PDF generating tool (page 3, section 3.3).

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Gebert in view of PODI to include reusable content objects are mapped to conventional workflows with variable data records for generating PDF content objects from PPML objects as taught by deBronkart, providing the benefit of promoting interoperability amongst printers allowing users to have a fast time to market (deBronkart, page 3, section 4) for variable data printing applications, allowing users to use desktop printers (deBronkart, Abstract), which is similar to the goal of the Applicant's invention of allowing users to use any printer such as low-end printers instead of high-end digital presses (see specification, page 1, para 3).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Gautam Sain whose telephone number is 571-272-4096. The examiner can normally be reached on M-F 9-5 EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Heather Herndon can be reached on 571-272-4136. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

GS 6/22/06
GS

HEATHER R. HERNDON
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100